



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

*m*

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/718,867	11/21/2003	Dong Ho Song	2101785-991110	1481

26379 7590 03/21/2007  
DLA PIPER RUDNICK GRAY CARY US, LLP  
2000 UNIVERSITY AVENUE  
E. PALO ALTO, CA 94303-2248

EXAMINER
----------

WANG, BEN C

ART UNIT	PAPER NUMBER
----------	--------------

2192

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	03/21/2007	PAPER

**Please find below and/or attached an Office communication concerning this application or proceeding.**

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

**Office Action Summary**

Application No.

10/718,867

Applicant(s)

SONG ET AL.

Examiner

Ben C. Wang

Art Unit

2192

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

- 1) ☒ Responsive to communication(s) filed on 21 November 2003.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

- 4) ☒ Claim(s) 1-46 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1-46 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

**Application Papers**

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on \_\_\_\_\_ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
  2. ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
  3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

- |   |   |
|---|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892)   | 4) <input type="checkbox"/> Interview Summary (PTO-413)<br>Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948)  | 5) <input type="checkbox"/> Notice of Informal Patent Application                       |
| 3) <input checked="" type="checkbox"/> Information Disclosure Statement(s) (PTO/SB/08)<br>Paper No(s)/Mail Date <u>08/26/2004; 08/30/2004</u> . | 6) <input type="checkbox"/> Other: _____  |

### **DETAILED ACTION**

1. Claims 1-46 are pending in this application and presented for examination.

#### ***Drawing Objections***

2. Drawing is objected to because the following informalities:
  - The labels of “No” and “Yes” associated with step “182” (conditional diamond box – “Is process list processed ID in OS process list”), Fig. 6, should be corrected as opposite way.

Appropriate correction is required.

#### ***Specification Objections***

3. The specification is objected to because the following informalities:
  - The acronyms of “SVCHOST”, cited in P. 2, Line 10, “VBX”, cited in P. 2, Line 14, “OLE”, cited in P. 13, Line 1, “GUID”, cited in P. 13, Line 12, and “SCM”, cited in P. 13, Line 13, should be spelled out once as the intended meaning of the term is likely to be changed over the time.
  - “Launcher will verify and establishes whether all the application”, cited in P. 22, line 10, should be corrected as “Launcher will verify and establish whether all the application”.

Art Unit: 2192

- "Figure 5 illustrates a process manager method in accordance with the invention", cited in P. 23, line 16, should be corrected as "Figure 6 illustrates a process manager method in accordance with the invention".

Appropriate correction is required.

### ***Claim Objections***

4. Claims 4-5 and 28-29 are objected to because the following informalities:
  - "repeating component hooker injection for all the new secured application process<sub>i</sub>", claim 4, line 56, should be corrected as "repeating component hooker injection for all the new secured application process<sub>1</sub>".
  - "for the termination of each said secured run-time application<sub>i</sub>", claim 5, line 64, should be corrected as "for the termination of each said secured run-time application<sub>1</sub>".
  - "repeating component hooker injection for all the new secured application process<sub>i</sub>", claim 28, line 221, should be corrected as "repeating component hooker injection for all the new secured application process<sub>1</sub>".
  - "for the termination of each said secured run-time application<sub>i</sub>", claim 29, line 229, should be corrected as "for the termination of each said secured run-time application<sub>1</sub>".

Appropriate correction is required.

***Claim Rejections – 35 USC § 102(e)***

5. The following is quotation of 35 U.S.C. 102(e) which form the basis for all obviousness rejections set forth in this office action:

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

6. Claims 1-7, 9-30, and 32-46 are rejected under 35 U.S.C. 102(e) as being anticipated by Blaser et al. (Pat. No. US 7,117,495 B2) (hereinafter 'Blaser')

7. **As to claim 1**, Blaser discloses a system for executing application software (Fig. 1, element 104 - Applications; Fig. 2, element 200 – Application) on a operating system within a secured run-time environment (Col. 11, Lines 20-38 – Secure Applications; Col. 11, Line 63 through Col. 12, Line 7 – Secure Base OS; Col. 13, Lines 1-4, 16-23) without affecting an application software resources of a client computer (Col. 7, Lines 24-34; Col. 13, Lines 30-33), the system comprising an application wrapper (Fig. 1, element 106 – Layering System Libraries/Software; Fig. 2, elements 202 – Layer B, 204 – Layer A; Fig. 3, elements 300 – FSL (File System Layer) Management Application, 306 – FSL API Library, 310 – FSL Compression Library, 312 – FSL System Driver), wherein the application wrapper shields the application software resources (Col. 4, Lines 15-26 – layering system libraries and/or software intercepts files system and registry access from applications), whereby the secured run-time environment for

executing the application software is created and the application software resources are protected (Col. 11, Lines 20-38 – Secure Applications; Col. 11, Line 63 through Col. 12, Line 7 – Secure Base OS; Col. 13, Lines 1-4, 16-23); and the application wrapper further comprising a privatized virtual file resource created from an operating system file system (i.e., Fig. 3; Col. 13, Line 58 through Col. 14, Line 11 – an FSL system driver is installed “on top” of the operating system in order to have first processing priority for file system accesses), a privatized virtual registry created from an operating system registry system (i.e., Fig. 3; Col. 13, Line 58 through Col. 14, Line 11 – an FSL system driver is installed “on top” of the operating system in order to have first processing priority for registry), a privatized operating system shared component resource (Col. 5, Lines 18-32; Col. 6, Line 62 through Col. 7, Line 17), a privatized application configuration resource (Col. 2, Lines 50-52 – having facilities for providing configuration settings to applications; Col. 3, Lines 42-46, 51-58; Col. 4, Lines 23-26 – a layer manager application may be provided to permit control and configuration of the layering system software through a management API and library; Col. 5, Lines 23-29; Col. 7, Lines 48-53; Col. 9, Lines 6-11; Col. 11, Lines 16-19) and a privatized environmental resources for application variables (Col. 8, Lines 48-61 – for example, Windows operating systems set the “WINDIR” environment variable to the location of the OS system subtree...; Col. 8, Line 67 through Col. 9, Line 11).

8. **As to claim 24**, Blaser discloses a method for executing application software on a operating system within a secured run-time environment (Col. 11, Lines 20-38 –

Art Unit: 2192

Secure Applications; Col. 11, Line 63 through Col. 12, Line 7 – Secure Base OS; Col. 13, Lines 1-4, 16-23) without affecting an application software resources of a client computer (Col. 7, Lines 24-34; Col. 13, Lines 30-33), the client compute comprising an application wrapper (Fig. 1, element 106 – Layering System Libraries/Software; Fig. 2, elements 202 – Layer B, 204 – Layer A; Fig. 3, elements 300 – FSL (File System Layer) Management Application, 306 – FSL API Library, 310 – FSL Compression Library, 312 – FSL System Driver), wherein the application wrapper shields the application software resources (Col. 4, Lines 15-26 – layering system libraries and/or software intercepts files system and registry access from applications), whereby a the secured run-time environment for executing an the application software is created and the application software resources is protected (Col. 11, Lines 20-38 – Secure Applications; Col. 11, Line 63 through Col. 12, Line 7 – Secure Base OS; Col. 13, Lines 1-4, 16-23), the method further comprising privatizing virtual file resource created from an operating system file system (i.e., Fig. 3; Col. 13, Line 58 through Col. 14, Line 11 – an FSL system driver is installed “on top” of the operating system in order to have first processing priority for file system accesses); privatizing virtual registry created from an operating system registry system (i.e., Fig. 3; Col. 13, Line 58 through Col. 14, Line 11 – an FSL system driver is installed “on top” of the operating system in order to have first processing priority for registry); privatizing operating system shared component resource (Col. 5, Lines 18-32; Col. 6, Line 62 through Col. 7, Line 17); privatizing application configuration resource (Col. 2, Lines 50-52 – having facilities for providing configuration settings to applications; Col. 3, Lines 42-46, 51-58; Col. 4, Lines 23-26 – a

Art Unit: 2192

layer manager application may be provided to permit control and configuration of the layering system software through a management API and library; Col. 5, Lines 23-29; Col. 7, Lines 48-53; Col. 9, Lines 6-11; Col. 11, Lines 16-19); and privatizing environmental resources for application variables (Col. 8, Lines 48-61 – for example, Windows operating systems set the "WINDIR" environment variable to the location of the OS system subtree...; Col. 8, Line 67 through Col. 9, Line 11).

9. **As to claims 2 and 25**, Blaser discloses privatized virtual file resource (i.e., Col. 16, Line 11 through Col. 18, Line 12 – File System Calls) further comprising intercepting file I/O request generated by one or more processes (Fig. 5, step 502 – "wait for read request"; Fig. 6, step 602 – "wait for write request"); establishing a process ID for the intercepted file I/O request (Col. 9, Lines 34-38, 58-63; Col. 30, Lines 52-53, 58-61); comparing process ID to establish operating system process and secured run-time process (Fig. 5, step 504 – "is file reference maintained in an enabled layer?"; Fig. 6, step 604 – "is file reference to be captured to an enabled layer?"; Col. 6, Lines 37-39; Col. 15, Lines 16-26); establishing a process ID as operating system process and secured run-time process (Col. 9, Lines 34-38, 58-63; Col. 30, Lines 52-53, 58-61); servicing the file I/O request for all process ID established as secured run-time process (Fig. 5, steps 508, 510, 512; Fig. 6, steps 608, 610, 612); redirecting the file I/O request to operating system service for process ID established as operating system process (Fig. 5, step 506 – "use reference of request to execute regular read"; Fig. 6, step 606 – "use reference of request to execute regular write"; Col. 4, Lines 15-26; Col. 9, Lines 29-



Art Unit: 2192

33; Col. 14, Line 64 through Col. 15, Line 3; Col. 15, Line 66 through Col. 16, Line 3; Col. 16, Lines 41-44; Col. 17, Lines 5-13, 24-28); rejecting the file I/O request on secured run-time process resources for process ID established as operating system process (Col. 4, Lines 15-26; Col. 9, Lines 29-33; Col. 14, Line 64 through Col. 15, Line 3; Col. 15, Line 66 through Col. 16, Line 3; Col. 16, Lines 41-44; Col. 17, Lines 5-13, 24-28); comparing process ID established as secured run-time process to further establish process resources corresponding to process ID (Col. 15, Lines 16-26); establishing corresponding process resources within secured run-time resources (Fig. 5, steps 508, 510, 512; Fig. 6, steps 608, 610, 612); and rejecting the file I/O request on secured run-time process resources for process ID established as secured run-time process and process ID belongs to other process resources (Col. 4, Lines 15-26; Col. 9, Lines 29-33; Col. 14, Line 64 through Col. 15, Line 3; Col. 15, Line 66 through Col. 16, Line 3; Col. 16, Lines 41-44; Col. 17, Lines 5-13, 24-28).

10. **As to claims 3 and 26**, Blaser discloses privatized virtual registry (i.e., Col. 18 Line 13 through Col. 20, Line 18 – Registry Calls) further comprises privatizing virtual registry system by intercepting registry I/O request generated by several process (Fig. 7, step 702 – “waiting for request for registry setting”; Col. 6, Lines 33-37); establishing process ID for the intercepted registry I/O request (Col. 9, Lines 34-38, 58-63; Col. 30, Lines 52-53, 58-61); comparing process ID to establish operating system process and secured run-time process (Fig. 7, step 704 – “is registry request to be captured to an enabled layer?”; Col. 6, Lines 37-39; Col. 15, Lines 16-26); establishing process ID as

Art Unit: 2192

operating system process and secured run-time process (Fig. 7, step 708 – identify destination layer; Col. 6, Lines 41-43); servicing the registry I/O request for all process ID established as secured run-time process (Fig. 7, elements 712 – create virtual registry setting in layer, 716- create delete entry in virtual registry, or delete entry if isolated to layer, 720 – set virtual registry setting in layer; Col. 6, Lines 44-55); redirecting the registry I/O request to operating system service for process ID established as operating system process (Fig. 7, element 706 – execute normal registry function; Col. 6, Lines 39-41; Col. 14, Line 64 through Col. 15, Line 3; Col. 15, Lines 16-26; Col. 15, Line through Col. 16, Line 11 – the structures and hooks); rejecting the registry I/O request on secured run-time process resources for process ID established as operating system process (Fig. 7, element 706 – execute normal registry function; Col. 6, Lines 39-41; Col. 15, Lines 16-26); comparing process ID established as secured run-time process to further establish process resources corresponding to process ID (Col. 15, Lines 16-26); establishing corresponding process resources within secured run-time resources (Fig. 7, elements 712, 716, 720); and rejecting the registry I/O request on secured run-time process resources for process ID established as secured run-time process and process ID belongs to other process resources (Fig. 7, elements 706 – execute normal registry function; Col. 6, Lines 39-41, 708 – identify destination layer; Col. 6, Lines 41-43).

11. **As to claims 4 and 28**, Blaser discloses privatizing operating system shared component resource (Col. 5, Lines 18-32; Col. 6, Line 62 through Col. 7, Line 17)

Art Unit: 2192

further comprising searching secured application process for injecting component hooker (Col. 9, Lines 43-46; Col. 15, Line 46 through Col. 16, Line 11); checking the secured application process to establish whether the process is injected with component hooker (Col. 16, Lines 36-38, 45-47); establishing the secured application process as new secured application process for the secured application process not injected with component hooker (Col. 8, Lines 11-18; Col. 19, Lines 28-36); injecting component hooker to new secured application process to intercept component process (Fig. 3, element 312 – FSL System Driver; Col. 9, Lines 6-11; Col. 13, Line 62 through Col. 14, Line 2; Col. 15, Lines 42-44 – all of the file system entry points are hooked in; the drives to be redirected are hooked in; all of the Registry entry points are hooked in); repeating component hooker injection for all the new secured application process (Col. 12, Lines 29-31).

12. **As to claims 5 and 29**, Blaser discloses privatizing operating system shared component resource further comprising Initializing component redirection table to provide component redirecting information (Fig. 7, element 712 – create virtual registry setting in layer; Col. 14, Lines 17-22, 30-31 – *FileRedirect*, 36-37 – *RegRedirect*; Col. 20, lower portion for Function/Description List – *FSLCreate()* – create the layer redirection area in the file system); Registering virtual component required for the secured application (Fig. 7, element 716 – set virtual registry setting in layer; Col. 27 – Function/Description List – *FSLSetLayerInfo()* – if *fileRedirect* is specified, set the value of the proper registry value... create the redirect root keys..); Adding redirecting

Art Unit: 2192

information to the component redirection table for the execution of each selected the secured run-time application (Col. 25 – Function/Description list –

*FSLRegCreateKeyEx()* – create a registry path to the layer's redirection area using the layer's redirect path, its name, .... Create the key in the redirection area); Removing component redirecting information from the component redirection table for the termination of each the secured run-time application (Fig. 7, element 716 – delete entry if isolated to layer).

13. **As to claims 6 and 27**, Blaser discloses privatizing operating system shared component resource further comprising intercepting component process function for replacing component search path with secured application resource path (Fig. 5, element 502; Fig. 6, element 602; Fig. 7, element 702); replacing component search path with private resource path to load the component from the secured application resource path (Col. 5, Lines 60-64; Col. 6, Lines 12-17; Col. 8, Lines 48-52); and creating new process for the intercepted component with the replaced secured application resource path (Col. 15, Lines 58-60; Col. 16, Lines 19-48; Col. 16, Line 53 through Col. 17, Line 36; Col. 18, Lines 26-61).

14. **As to claims 7 and 30**, Blaser discloses privatizing operating system shared component resource further comprising intercepting component call for replacing component registry path with the privatized virtual registry path (Fig. 7, element 702); searching component redirection table for the component redirecting information (Col.

Art Unit: 2192

18, Lines 26-28; Col. 19, Lines 24-26; Col. 19, Line 37 through Col. 20, Line 3); replacing component registry path with the privatized virtual registry path retrieved from the component redirection table (Col. 15, Lines 58-60; Col. 16, Lines 19-48; Col. 16, Line 53 through Col. 17, Line 36; Col. 18, Lines 26-61); returning the intercepted call to the requested call with the replaced secured application registry path for addressing the component location from the privatized virtual registry system and further the component is addressed to load from the privatized virtual file system (Fig. 7, element 708; Col. 6, Lines 41-43); redirecting the component call as it is to the requested call for the component call originated from non secured run-time application and for the component call, which do not have redirecting information in the component redirection table (Fig. 7, element 706 – “execute normal registry function”; Col. 6, Lines 39-41).

15. **As to claims 9 and 32**, Blaser discloses privatized application configuration resource further comprises monitoring file I/O request for configuration file to provide separate configuration file (Fig. 5, element 502; Fig. 6, element 602; Col. 9, Lines 6-11); searching and retrieving configuration file from secured application resources (Col. 2, Lines 50-52 – having facilities for providing configuration settings to applications; Col. 3, Lines 42-46, 51-58; Col. 4, Lines 23-26 – a layer manager application may be provided to permit control and configuration of the layering system software through a management API and library); and serving application configuration file to requesting process (Col. 5, Lines 23-29; Col. 7, Lines 48-53; Col. 11, Lines 16-19).

16. **As to claims 10 and 33**, Blaser discloses privatized environmental resources (Col. 8, Lines 48-61 – for example, Windows operating systems set the "WINDIR" environment variable to the location of the OS system subtree...; Col. 8, Line 67 through Col. 9, Line 11) further comprises intercepting environment variable request to supply private values to secured application process (Fig. 5, element 502; Fig. 6, element 602; Col. 9, Lines 6-11); verifying process ID to establish the process ID as operating system process or secured application process (Fig. 5, step 504 – "is file reference maintained in an enabled layer?"; Fig. 6, step 604 – "is file reference to be captured to an enabled layer?"; Col. 6, Lines 37-39; Col. 15, Lines 16-26); redirecting the call for process ID established as operating system process (Fig. 5, element 506; Fig. 6, element 606); reading variable data from secured application resource and returning the value to requested process for read variable calls requested by the secured application process (Fig. 5, element 512; Col. 5, Lines 60-64) ; searching the requesting write variable in secured application resource to find the presence of requesting write variable (Col. 23 – Function/Description List – *FSLFindCloseVariable()*, *FSLFindFirstVariable()*, *FSLFindNextVariable()*, *FSLGetVariable()*); creating variable with variable data within secured application resource and returning the status to requested process for variable do not exist in secured application resource (Col. 20, lower portion – Function/Description List – *FSLAddVariable()*); and updating variable with variable data within secured application resource and returning the status to requested process for variable exist in secured application resource (Col. 8, Line 61 through Col. 9, Line 4).

17. **As to claims 11 and 34**, Blaser discloses the application wrapper further comprises selectively allowing the application software to interact operating system resources directly during the application software executing under the secured run-time environment (Fig. 3, element 304 – Other Applications; Col. 14, Lines 6-11).

18. **As to claims 12 and 35**, Blaser discloses the application Wrapper further comprises selectively allowing the application software to interact with other application software resources directly during the application software executing under the secured run-time environment (Fig. 3, element 304 – Other Applications; Col. 14, Lines 6-11).

19. **As to claims 13 and 36**, Blaser discloses the application wrapper further comprises providing a run-time environment to the application software that is visible to an operating system run-time environment without having the application software run-time resources, whereby the application software resources is simulated to the secured run-time environment during the execution of the application software (Col. 12, Lines 8-20 – all further changes are recorded to a layer and not to the underlying file systems and OS resources).

20. **As to claims 14 and 37**, Blaser discloses the system and method further comprising means (Fig. 1, element 106 – Layering System Libraries/Software; Fig. 2, elements 202 – Layer B, 204 – Layer A; Fig. 3, elements 300 – FSL (File System Layer)

Art Unit: 2192

Management Application, 306 – FSL API Library, 310 – FSL Compression Library, 312 – FSL System Driver) for protecting the behavior of the application software from other application and the operating system (Col. 11, Lines 20-38 – Secure Applications; Col. 11, Line 63 through Col. 12, Line 7 – Secure Base OS; Col. 13, Lines 1-4, 16-23).

21. **As to claims 15 and 38**, Blaser discloses the system and method further comprising means (Fig. 1, element 106 – Layering System Libraries/Software; Fig. 2, elements 202 – Layer B, 204 – Layer A; Fig. 3, elements 300 – FSL (File System Layer) Management Application, 306 – FSL API Library, 310 – FSL Compression Library, 312 – FSL System Driver) for eliminating the application conflicts from other running application software (Col. 2, Lines 42-46; Col. 3, Lines 51-58; Col. 8, Lines 11-18).

22. **As to claims 16 and 39**, Blaser discloses the system and the method further comprising means for executing multiple instance of the single application software (Col. 8, Lines 6-11).

23. **As to claims 17 and 40**, Blaser discloses the system and the method wherein the application wrapper further comprising maintaining the application software resources away from the operating system resources (Col. 4, Lines 15-26 – layering system libraries and/or software intercepts files system and registry access from applications), whereby the operating system resources is protected from the application



Art Unit: 2192

software resources (Col. 11, Lines 20-38 – Secure Applications; Col. 11, Line 63 through Col. 12, Line 7 – Secure Base OS; Col. 13, Lines 1-4, 16-23).

24. **As to claims 18 and 41**, Blaser discloses the system and the method wherein the application wrapper further comprises permitting full access to the application software that requires to access for variation occurs to the application software resources within the application wrapper (Col. 4, Lines 15-26; Col. 8, Lines 19-23, 48-55).

25. **As to claims 19 and 42**, Blaser discloses the system and the method further comprising means for keeping the state of secured run-time environment to the application software (Col. 6, Lines 3-10; Col. 7, Lines 9-12; Col. 10, Lines 2-24; Col. 11, Line 63 through Col. 12, Line 7).

26. **As to claims 20 and 43**, Blaser discloses the system and the method further comprising means for updating the application software resources required by the application software (Col. 5, Lines 18-32 - shared libraries, system accessible configuration, and version control are managed by the layering subsystem).

27. **As to claims 21 and 44**, Blaser discloses the system and the method wherein the application wrapper monitors the application run-time request to determine the

Art Unit: 2192

required the application software resources for execution (Fig. 1, element 106 – Layering System Libraries/Software; Col. 4, Lines 15-26).

28. **As to claims 22 and 45**, Blaser discloses the system and the method further comprising means for receiving the application software resources to execute the application software in the secured run-time environment (Fig. 1, element 106 – Layering System Libraries/Software; Fig. 2, elements 202 – Layer B, 204 – Layer A; Fig. 3, elements 300 – FSL (File System Layer) Management Application, 306 – FSL API Library, 310 – FSL Compression Library, 312 – FSL System Driver).

29. **As to claims 23 and 46**, Blaser discloses the system and the method further comprising means for incrementally executing the application software in the secured run-time environment (Col. 3, Line 59 through Col. 4, Line 6; Col. 5, Lines 18-32 – shared libraries, system accessible configuration, and version control are managed by the layering subsystem).

### ***Claim Rejections – 35 USC § 103(a)***

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made

30. Claims 8 and 31 are rejected under 35 U.S.C. 103(a) as being unpatentable over Blaser in view of Osman et al., (*The Design and Implementation of Zap: A System for Migrating Computing Environments*, 2002, ACM, pp. 361-376) (hereinafter 'Osman') and in further view of G. Hunt (Pub. No. US 2002/0072830 A1) (hereinafter 'Hunt')

31. **As to claims 8 and 31**, although Blaser discloses hooking/redirectioned structures (Col. 15, Line 38 through Col. 20, Line 18) and its intercept/redirectioned mechanism for file system and registry accesses from applications (Col. 4, Lines 15-26), but does not explicitly disclose privatizing operating system shared component resource further comprising intercepting the RPC message call for replacing component information with privatized virtual component information; searching component redirection information from the component redirection table; replacing RPC message with the privatized virtual component information retrieved from the component redirection table; returning the intercepted RPC message call to the requested call with the replaced message; continuing the RPC call to locate the privatized virtual component through SCM; redirectioned the RPC message call as it is to the requested call for the component call originated from non secured run-time application and for the component call, which do not have redirectioned information in the component redirection table.

However, in an analogous art of a system for migrating computing environments, Osman discloses intercepting the RPC message call for replacing component information with privatized virtual component information (Table 1 – Pod principles applied to resources – Network Resource; Sec. 4 – Zap Architecture, 2<sup>nd</sup> Para., Lines 1-

Art Unit: 2192

12; Sec. 4.5 – Network Virtualization and Migration, 1<sup>st</sup> Para., Lines 4-11; 8<sup>th</sup> Para., Lines 1-8 – connection virtualization works by intercepting system calls for connection setup requests from the application to the transport protocol and replacing relevant physical addresses with virtual address); searching component redirecting information from the component redirection table; replacing RPC message with the privatized virtual component information retrieved from the component redirection table returning the intercepted RPC message call to the requested call with the replaced message (Sec. 4.5 – Network Virtualization and Migration, 9<sup>th</sup> Para., Lines 1-9 – Connection translation works by intercepting packets below the transport protocol layer and translating the virtual address in the packet headers to physical address).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Osman into the Blaser's system to further provide intercepting the RPC message call for replacing component information with privatized virtual component information; searching component redirecting information from the component redirection table; replacing RPC message with the privatized virtual component information retrieved from the component redirection table; returning the intercepted RPC message call to the requested call with the replaced message in Blaser's system.

The motivation is that it would enhance the Blaser's system by taking, advancing and/or incorporating Osman's system which provides a framework for network virtualization as once suggested by Osman (i.e., Sec. 4.5 – Network Virtualization and Migration, 1<sup>st</sup> Para.).

Although Osman discloses network virtualization (Sec. 4.5) but both Blaser and Osman do not explicitly disclose continuing the RPC call to locate the privatized virtual component through SCM; redirecting the RPC message call as it is to the requested call for the component call originated from non secured run-time application and for the component call, which do not have redirecting information in the component redirection table.

However, in an art of dynamic classification of sections of software, Hunt discloses the protocol stack, protocol information for remote communication, the DCOM network protocol which is a superset of Open Group's Distributed Computing Environment Remote Procedure Call (DCE RPC) protocol, and Service Control Manager; All of them are the essential building blocks for distributed Windows applications ([0076], Lines 21-33; Fig. 4, elements 108/144 – Service Control Manager, 120 – DCOM Network Protocol).

Further Hunt discloses continuing the RPC call ([0077], Lines 1-6, 21-33 – the proxy and stub typically include a protocol stack and protocol information for remote communication, for example, the DCOM network protocol, which is a superset of the Open Group's Distributed Computing Environment Remote Procedure Call (DCE RPC) protocol) to locate the privatized virtual component through SCM; redirecting the RPC message call as it is to the requested call for the component call originated from non secured run-time application and for the component call, which do not have redirecting information in the component redirection table (Fig. 4, elements 108, 144 – Service Control Manager; [0073] – after instantiation, DCOM supports cross-process or cross-

Art Unit: 2192

machine communication; [0074] – a local service control manager connects to a remote service control manager, which requests creation of the component through the “CoCreateInstance” API).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Hunt into the Blaser-Osman system to further provide continuing the RPC call to locate the privatized virtual component through SCM; redirecting the RPC message call as it is to the requested call for the component call originated from non secured run-time application and for the component call, which do not have redirecting information in the component redirection table in Blaser-Osman’s system.

The motivation is that it would enhance the Blaser-Osman’s system by taking, advancing and/or incorporating Hunt’s system which is coupled with a distributed object system such as Microsoft™ Corporation’s Distributed Component Object Model (DCOM™) to further provide system services that support execution of distributed application once suggested by Hunt (i.e., [0005]).

**Conclusion**

32. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- Blaser et al., *Layered Computing Systems and Methods* (Pat. No. US 7,162,724 B2)
- Heidemann, et al., File System Development with Stackable Layers, 1993, ACM, pp. 1-20
- I. Ivanov, *API hooking revealed, April, 2002, the code project*, pp. 1-29
- K. Newcomb, Softricity Has Cure for App Conflict Blues, May 2002, Thin Planet website, pp. 1-6

33. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.


Art Unit: 2192

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

BCW

*PW*

March 14, 2007

  
SUPERVISOR